Listing of Claims

1-66. (Cancelled)

67. (Previously presented) A computer implemented method for adding tamper resistance to a software program, the method comprising:

adding a silent guard variable to the software program;

selecting a computation in the software program;

determining an expected value of the silent guard variable at the execution point of the selected computation;

setting the runtime value of the silent guard variable to the expected value of the silent guard variable in the software program at a silent guard insertion point, the silent guard insertion point being separated from the execution point of the selected computation by a plurality of program instructions of the software program; and

revising the selected computation to be dependent on the runtime value of the silent guard variable, such that the software program executes improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.

68. (Previously presented) The method of claim 67, further comprising:

selecting a program variable in the software program;

determining an expected value of the selected program variable at a dependency point in the software program; and

making the runtime value of the silent guard variable dependent on the runtime value of the selected program variable at the dependency point.

69. (Previously presented) The method of claim 68, wherein the step of making the value of the silent guard variable dependent on the runtime value of the selected program variable comprises:

computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point.

70. (Previously presented) The method of claim 67, wherein the step of revising the selected computation comprises:

selecting a constant value used in the selected computation; and

replacing the constant value with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable.

71. (Previously presented) The method of claim 67, wherein the step of revising the selected computation comprises:

selecting a computation variable used in the selected computation;

determining an expected value of the computation variable at the execution point of the selected computation; and

replacing the computation variable with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression

evaluates to the expected value of the computation variable if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable.

72. (Previously presented) The method of claim 67, wherein the step of revising the selected computation comprises:

inserting a mathematical expression including the runtime value of the silent guard variable and the expected value of the silent guard variable into the selected computation.

73-77. (Canceled)

78. (Previously presented) A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program variable in the software program;

selecting a computation in the software program;

determining an expected value of the program variable at the point of execution of the selected computation; and

revising the selected computation to be dependent on the runtime value of the program variable, such that the software program executes incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable.

79. (Previously presented) The method of claim 78, wherein the step of revising the selected computation comprises:

selecting a constant value used in the selected computation; and

replacing the constant value with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the constant value if the runtime value of the program variable is equal to the expected value of the program variable.

80. (Previously presented) The method of claim 78, wherein the step of revising the selected computation comprises:

selecting a computation variable used in the selected computation;

determining an expected value of the computation variable at the point of execution of the selected computation; and

replacing the computation variable with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the program variable is equal to the expected value of the program variable.

81. (Previously presented) The method of claim 78, wherein the step of revising the selected computation comprises:

inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation.

82. (Currently amended) A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program block ~~containing a step~~ that computes a result necessary for

proper execution of the software program, the program block comprising at least one program instruction;

selecting a silent guard for the program block;

determining the expected value of the silent guard at the start of execution of the program block; and

installing a ~~branch instruction~~ first computation dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the ~~branch instruction~~ first computation causes the result computed by the program block to ~~be skipped~~ evaluate improperly, causing the software program to execute improperly.

83. (Currently amended) The method of claim 82, wherein the step of selecting a silent guard comprises:

adding a silent guard variable to the software program;

using the silent guard variable as the silent guard; and

installing an initialization instruction for the silent guard variable that executes prior to the ~~branch instruction~~ first computation in the software program, the initialization instruction setting the silent guard variable equal to the expected value of the silent guard.

84. (Previously presented) The method of claim 82, wherein the step of selecting a silent guard comprises:

selecting a program variable in the software program; and

using the program value as the silent guard.

85. (Previously presented) The method of claim 82, wherein the step of selecting a silent guard comprises:

selecting an insertion point in the software program;

selecting a program variable in the software program;

determining the expected value of the program variable at the insertion point;

making the runtime value of the silent guard dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point.

86. (Currently amended) The method of claim 85, wherein the step of making the runtime value of the silent guard dependent on the runtime value of the program variable, comprises:

installing a ~~mathematical~~ second computation that includes the runtime value of the program variable, such that the result of the ~~mathematical~~ second computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point; and

setting the silent guard equal to the result of the ~~mathematical~~ second computation.

87. (Previously presented) The method of claim 86, wherein the silent guard computation uses both the expected value of the program variable and the runtime value of the program variable.

88. (Previously presented) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program computation at an execution point in the software program;

a silent guard variable having an expected value at the execution point;

a program variable having an expected value at a dependency point in the software program, the dependency point being separated from the execution point by a plurality of program instructions, the runtime value of the silent guard variable being dependent on the runtime value of the program variable at the dependency point, such that the runtime value of the silent guard variable will not equal the expected value of the silent guard variable at the execution point if the runtime value of the program variable is not equal to the expected value of the program variable at the dependency point; and

wherein the program computation is dependent on the runtime value of the silent guard variable, such that the program computation will evaluate improperly and the software program will execute improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.

89. (Canceled)

90. (Previously presented) The recordable computer media of claim 88, wherein the runtime value of the silent guard variable is also dependent on the expected value of the program variable at the dependency point.

91. (Previously presented) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program variable having an expected value at a first dependency point in the software program;

a silent guard variable having an expected value at the first dependency point;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

wherein the runtime value of the program variable is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly.

92. (Previously presented) The recordable computer media of claim 91, the tamper resistant software program further comprising an instruction setting the value of the program variable at the first dependency point equal to a function of the runtime value of the program variable and the result of the mathematical computation.

93. (Cancelled)

94. (Currently amended) ~~The~~ A recordable computer media ~~of claim 91~~ having a tamper resistant software program recorded thereon, the tamper resistant software program ~~further~~ comprising~~.~~:

a program variable having an expected value at a first dependency point in the software program and an expected value at a second dependency point in the software program, the expected value at the first dependency point not being equal to the expected value at the second dependency point;

a silent guard variable having an expected value at the first dependency point;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

a supplementary silent guard variable having an expected value at ~~a~~ the second dependency point in the software program;

wherein the runtime value of the program variable is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly; and

wherein ~~the expected value of the program variable at the second dependency point is not equal to the expected value of the program variable at the first dependency point; and~~ the runtime value of the program variable at the second dependency point is dependent on the runtime value of the supplementary silent guard variable, such that the

runtime value of the program variable will not equal the expected value of the program variable at the second dependency point if the runtime value of the supplementary silent guard variable does not equal the expected value of the supplementary silent guard variable at the second dependency point, which will cause the software program to execute improperly.

95. (Currently amended) ~~The~~ A recordable computer media ~~of claim 91,~~ having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program variable having an expected value at a first dependency point in the software program and an expected value at a second dependency point in the software program, the expected value at the first dependency point not being equal to the expected value at the second dependency point;

a silent guard variable having an expected value at the first dependency point and an expected value at the second dependency point in the software program;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

wherein the runtime value of the program variable is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly; and

wherein ~~the expected value of the program variable at a second dependency point is not equal to the expected value of the program variable at the first dependency point; and the silent guard variable has a second expected value at the second dependency point; and~~ the runtime value of the program variable at the second dependency point is dependent on the runtime value of the silent guard variable at the second dependency point, such that the runtime value of the program variable will not equal the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the second dependency point, which will cause the software program to execute improperly.

96-100. (Cancelled)

101. (Previously presented) The method of claim 78, wherein the program variable is used in more than one instruction of the software program, and the last instruction to use the program variable before the point of execution of the selected computation during execution of the software program is separated from the point of execution of the selected computation by a plurality of program instructions not using the program variable.

102. (Previously presented) The method of claim 83, wherein the installation point of the initialization instruction for the silent guard variable is separated from the installation point of the branch instruction by a plurality of program instructions of the software program.

103. (Previously presented) The method of claim 85, wherein the insertion point is separated from

the installation point of the branch instruction by a plurality of program instructions of the software program.

104. (Previously presented) The recordable computer media of claim 94, wherein the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program.

105. (Previously presented) The recordable computer media of claim 95, wherein the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program.

106. (Previously presented) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a silent guard variable having an expected value at a first dependency point in the software program;

a program variable having an expected value at a second dependency point in the software program;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

wherein the runtime value of the program variable at the second dependency point is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable at the first dependency point, such that the runtime value of

the program variable at the second dependency point will not equal the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable at the first dependency point does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly.

107. (Previously presented) The recordable computer media of claim 106, wherein the first dependency point is separated from the second dependency point by a plurality of program instructions.

108. (Previously presented) The recordable computer media of claim 106, the tamper resistant software program further comprising an instruction setting the value of the program variable at the second dependency point equal to a function of the runtime value of the program variable and the result of the mathematical computation.

109-110. (Cancelled)

111. (Currently amended) The recordable computer media of claim ~~110~~ 112, wherein the insertion point is separated from the branch instruction by a plurality of program instructions.

112. (Currently amended) ~~The recordable computer media of claim 110,~~ A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program block for causing improper execution of the software program, the program block including at least one program instruction;

a silent guard in the software program having an expected value at the start of execution of the program block;

a program variable having an expected value at an insertion point, wherein the runtime value of the silent guard is made dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, ~~wherein~~ the result of the mathematical computation ~~is~~ being corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point;

a branch instruction in the software program dependent on the runtime value of the silent guard, wherein the branch instruction will cause the program block to be executed if the runtime value of the silent guard is not equal to the expected value of the silent guard, causing the program to execute improperly.

113. (Previously presented) The recordable computer media of claim 112, wherein the mathematical computation includes an expected term, the expected term being set based on the expected value of the program variable at the insertion point.

114. (Currently amended) A method for adding tamper resistance to a software program, the method comprising:

~~adding~~ determining a program block in the software program having a program variable with an expected value that causes improper execution of the software program when the program variable is not equal to the expected value of the program variable, the program block including at least one program instruction;

inserting a ~~branch instruction~~ silent guard having an expected value at the start of execution of the program block;

making the ~~branch instruction~~ program variable dependent on ~~a~~ the silent guard using a mathematical function ~~having an expected value and a runtime value~~, such that the ~~branch instruction causes the program block to be executed~~ mathematical function does not compute the expected value of the program value if the silent guard is not equal to the expected value of the silent guard, which causes the software program to execute improperly.

115. Cancelled

116. (Currently amended) The method of claim ~~115~~ 117, wherein the insertion point is separated from the branch instruction by a plurality of program instructions.

117. (Currently amended) ~~The method of claim 115, wherein~~ A method for adding tamper resistance to a software program, the method comprising:

adding a program block that causes improper execution of the software program, the program block including at least one program instruction;

inserting a branch instruction at the start of execution of the program block;

identifying a program variable in the software program having an expected value

at an insertion point;

making the branch instruction dependent on a silent guard having an expected value and a runtime value;

making the runtime value of the silent guard ~~is made~~ dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, wherein the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point, and the branch instruction causes the program block to be executed if the runtime value of the silent guard is not equal to the expected value of the silent guard.

118. (Previously presented) The method of claim 117, wherein the mathematical computation includes an expected term, the expected term being set based on the expected value of the program variable at the insertion point.

119. (New) The method of claim 114, wherein the mathematical computation includes the runtime value of the silent guard and the expected value of the silent guard.